TIPOS DE MEMORIAS Y QUE SE ALMACENA EN CADA UNA DE ELLAS

en Java Virtual Machine

Diego Yael Dorantes Rodriguez

Axel Reyes Guevara

MEMORIA DE PROGRAMA (CODE MEMORY):

Almacena el código de la aplicación Java compilada. Esta memoria es de sólo lectura y se utiliza para la ejecución de la aplicación.

MEMORIA HEAP (HEAP MEMORY):

Es el área donde se almacenan los objetos creados por la aplicación Java en tiempo de ejecución. Esta memoria es administrada automáticamente por el recolector de basura de Java.

MEMORIA PERMGEN (PERMANENT GENERATION):

Es una sección de la memoria de la JVM que almacena metadatos sobre las clases y los métodos que se utilizan en la aplicación. A partir de Java 8, esta memoria fue eliminada y se reemplazó por el espacio de metadatos (Metaspace).

MEMORIA STACK (STACK MEMORY):

Cada hilo de la aplicación Java tiene su propia pila de ejecución. La pila almacena información sobre los métodos que se están ejecutando actualmente y las variables locales asociadas.

MEMORIA NATIVE (NATIVE MEMORY):

Es la memoria utilizada por la JVM para almacenar información no gestionada por Java. Esto puede incluir bibliotecas nativas, archivos mapeados en memoria, entre otros.

Explicación y ejemplo del uso de las memorias

Ejemplo

```
public static boolean esPalindromo(String texto, int i){
      int largo = texto.length();
      int mitad = largo/2;
      if(i<mitad && texto.charAt(i) == texto.charAt((largo-1)-i))
         return esPalindromo(texto, i+1);
      return (i==mitad);
   }
```

public static boolean esPalindromo(String texto, int i){
int largo = texto.length();
int mitad = largo/2;

Recibe una palabra junto con un valor 0 que se utiliza como un contador
Después ala variable largo se le asigna la cantidad de caracteres de la palabra, a la variable mitad le asigna la mitad de los caracteres totales de la palabra.

if(i<mitad && texto.charAt(i) == texto.charAt((largo-1)-i))

Se compara que el contador sea menor que la variable mitad y que la letra en la posición i de la palabra sea igual a la invertida de la posición i de la palabra.

return esPalindromo(texto, i+1);
return (i==mitad);

En el caso que sea verdadero retornara el llamado a si mismo incrementando la variable i de la palabra.

En caso de que sea falso retornara true o false dependiendo de si la i es igual a la variable de la mitad.

Siendo i igual a mitad significa que todas las condiciones hasta la mitad de la palabra fueron iguales dando a entender que la palabra se escribe de la misma forma como de izquierda a derecha y viceversa demostrando que la palabra es un palindromo.